05 月 03 日

第二章：从素材到发布，用 Ulysses 高效写作

| 本文为付费栏目文章，您已订阅，可阅读全文 |

比起绘画、音乐和电影等其它艺术门类，写作所需的仅是一张纸，一支笔。从设备的复杂度上来说，可以说简单极了。

但这不意味着写作的流程也很简单。收集整理素材，为行文做好铺垫；合理地组织文章结构，以保持上下文的通顺；初稿完成后还要小心翼翼地润色。若是抱着「拿支笔就可以开始写」的想法，怕是很容易就陷入无法动笔的困境。

在第一章，Ulysses 沉浸而流畅的书写体验有所展现，但还没深入到真正写作的环境中。在这一章中，便来看看**从素材到写作**，**从编辑到发布**，Ulysses 作为你的「数字书桌」，如何让我们高效地完成写作。

用 Ulysses 收集素材

不管是科技话题的说明文，还是非虚构类特稿，乃至完全虚构的小说，收集素材永远都是继文章大体构思后，该进行的第一步。

所谓「素材」，其来源也非常灵活。既有自己脑中一闪而过的灵感，也有摘抄自网页的内容。如果你习惯使用印象笔记等工具，其中收集的内容也会成为素材的来源之一。

而 Ulysses 作为出色的写作工具，将这些需求都已经考虑在内。

用「智能粘贴」来摘抄

在为少数派写作时，我用到的有用素材几乎都来自互联网。屏幕左边放着浏览器，右边开着 Ulysses，是写作时的标配动作。

当复制粘贴网页上的内容到 Ulysses 中时，会出现一个问题：网页上的文字本质上是 HTML，而 Ulysses 则是纯文本，简单粘贴过来的内容失去了网页上包含的内容，例如链接等。更不方便的是，如果要拷贝代码这些内容，纯文本就看上去特别简陋。虽说之后可以在 Ulysses 中调整，但在收集素材时就花费太多时间，实在不得当。

Ulysses 解决这个需求的办法就是「**智能粘贴**」。

当从网页拷贝内容后，在 Ulysses 中不要简单地按 `⌘Command + V`，而是 `⌘Command + ⇧Shift + V`（或者长按 `⌘Command + V`），就能呼出智能粘贴菜单。最常用的是选择粘贴为 HTML 格式，网页中附带的链接等信息就会自动保存下来。

使用智能粘贴将网页内容拷贝为 HTML

这个功能看似不起眼，体验却好到能激发你用 Ulysses 来收集。拷贝时不用纠结网页上内容的格式，粘贴时保证一步到位且信息不丢失，又能让收集的素材都保持高颜值。实现了边阅读网页边摘抄的高效操作。如果没有这个功能，就会因为操作繁琐又不易阅读，让我想找个替代品来完成收集这个动作。

从印象笔记和备忘录中导入

说到收集就绕不开印象笔记这款应用。可以不夸张的说，互联网上有文字的地方，就有办法用印象笔记收集。

对于已经习惯使用印象笔记的用户，Ulysses 自然会加以考虑。将印象笔记中的笔记导入到 Ulysses 非常简单 ，在印象笔记中**将文件导出**为 `enex` 格式的文件，再**直接拖拽到 Ulysses 的文稿栏**中。

导入印象笔记

不过由于 `enex` 本质上是富文本格式的文件，所以导入 Ulysses 后有几点需要注意：

- 导入后 Ulysses 以 Markdown 的方式表示文章的原有样式，所以某些特殊样式可能不能正确显示；
- Ulysses 不支持表格和待办清单的语法标记，所以网页中的表格和清单会以纯文本出现；
- Ulysses 编辑界面中只有一种字体，网页上的多种字体会被忽视；

- 有密码保护的笔记不会被导入。

收集网页内容是印象笔记的强项，而到了收集随手记下的灵感和笔记的场景，我想不少人会使用苹果自带的备忘录应用。特别是 iOS 11 后可以在控制中心直接打开备忘录，使得它成为最便捷的灵感收集工具。

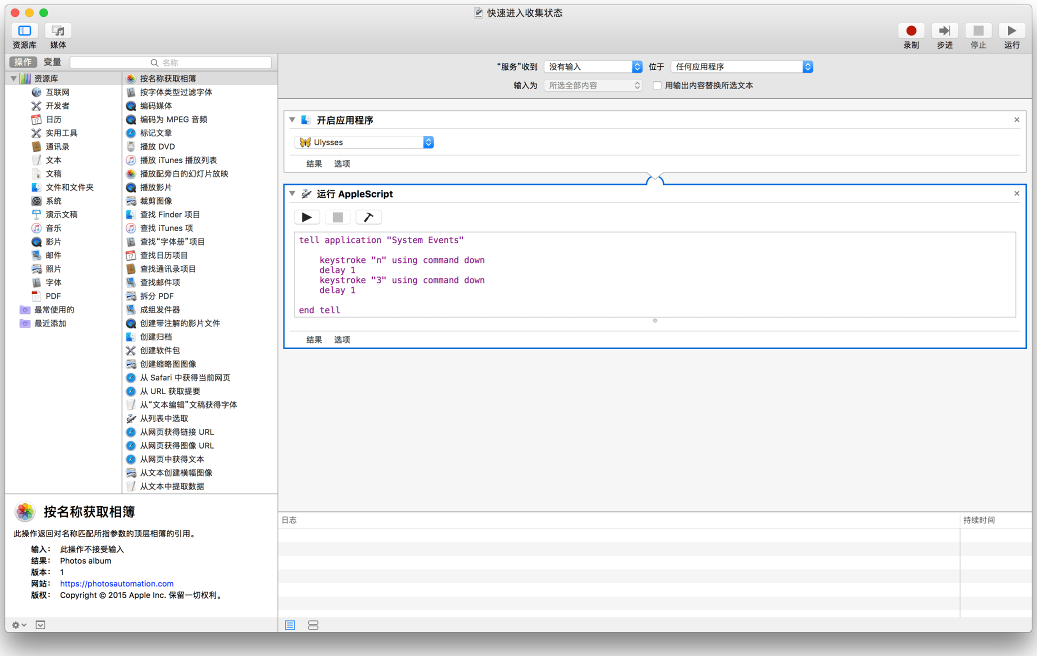
从备忘录导入

因为备忘录只能把笔记导出为 PDF，所以不能像印象笔记一样导出文稿再导入。Ulysses 特别开发了一款导入辅助工具（在 Ulysses 官方博客中下载）。工具的功能和使用非常简单，点击「Start」后会自动将备忘录中所有的笔记（同印象笔记一样，也不包括有密码保护的笔记）导入，存放在 Ulysses 中一个新建的「Imported from Notes」组中。

快速进入收集状态

在浏览网页时，发现有些内容特别适合作为素材，就想收集起来，日后可能会用得上。

而 Ulysses 没有设计菜单栏小工具，也没有全局新建文稿快捷键，就得靠我们自己 Hack 出一个快速进入 Ulysses 收集状态的办法。



“服务”收到 没有输入 位于 任何应用程序

输入为 所选全部内容 ☐ 用输出内容替换所选文本

开启应用程序

Ulysses

运行 AppleScript

```
tell application "System Events"

    keystroke "n" using command down
    delay 1
    keystroke "3" using command down
    delay 1

end tell
```

按名称获取相簿

此操作返回对名称匹配所指参数的顶层相簿的引用。

输入： 此操作不接受输入
结果： Photos album
版本： 1
网站： https://photosautomation.com
版权： Copyright © 2015 Apple Inc. 保留一切权利。

进入收集状态的 Automator 服务

为了随时能打开 Ulysses 且进入收集状态，就需要将 「打开 Ulysses」、「新建文稿」等一串动作连贯起来，用一个全局快捷键唤起。我使用的方案是借助 macOS 自带的 Automator 完成。思路其实很直接（点此下载 Automator 服务）：

1. 首先激活 Ulysses；
2. 然后在使用快捷键「⌘Command + N」新建文稿；

3. 最后将 Ulysses 设置为「仅限编辑器」（快捷键 `⌘Command + 3`）视图。

第一步使用 Automator 内部提供的操作完成，第二、三则用到了 AppleScript。

下载好提供的文件后双击安装。为了能通过系统全局快捷键随时使用它，在「系统偏好设置 – 键盘 – 快捷键 – 服务 – 通用」中为它分配一个快捷键，就可以随时启动这个 Automator 服务。

快速进入收集状态

深入了解 Ulysses 中的标记语法

使用 Ulysses 写作过程中，对语法标记的使用是关键。在之前的教程中，我们都以「在 Ulysses 中使用 Markdown XL 语法」为前提，对文稿的使用，都限于 Ulysses 之内。

可惜的是，Markdown XL 并不能被其他编辑器或者网络平台正确预览，或导出 HTML 或者其它格式文件。要将 Markdown 文稿运用到其它平台（例如 GitHub）时，就需要使用平台对应的 Markdown 语法。

在这个过程中，需要学会如何**自定义语法标记**，以及了解 Ulysses **导入导出文稿时的语法转换机制**。

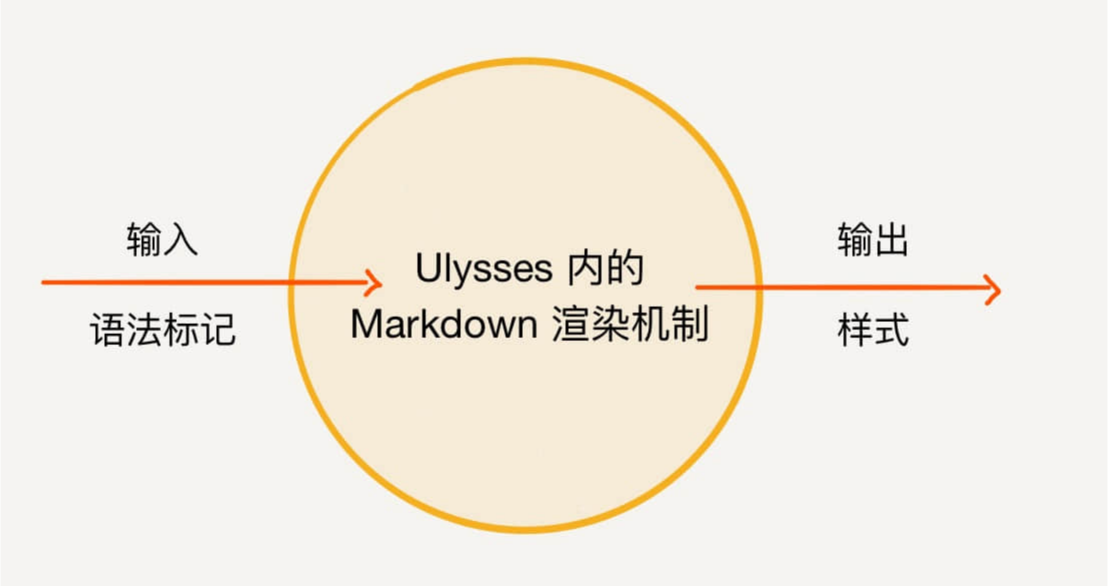
Ulysses 中标记语法的本质

在学习自定义语法标记前，我们先回顾下第一章中所说的 Markdown 的本质：

> 用语法标记来注明文字的样式，之后由软件（例如 Ulysses），将夹带着语法标记的文稿转换成富文本。

想让 Markdown 软件顺利工作，其中的关键就在于**标记和样式的一一对应**。

例如 Markdown XL 文本中输入 `#`，Ulysses 就知道「哦！他想把后面的字符当作一级标题」。这种对语法标记的理解能力，是开发者在程序内部实现的。


输入
语法标记
Ulysses 内的
Markdown 渲染机制
输出
样式

Ulysses 的作用

通常来说，一款 Markdown 的编辑器会支持一套标记语法，标记和样式之间的对应关系在程序中是固定的。而 Ulysses 则实现了所谓的**「360 度语义标记」**。

何谓「360 度语义标记」呢？再回到「`#` 对应一级标题」这个例子。在 Markdown XL 中我想写个一级标题，用 `#` 表示就可以了。

但是为什么一定要用 `#` 呢？说到底这只是一种约定俗成。本质上用任何符号，例如 `@` 来表示也可以。这就是「360 度语义标记」的第一层含义：**我们可以在 Ulysses 中自由地（360 度）修改某个语法标记的符号。**

「360 度语义标记」的第二层含义体现在导出上。依然是一级标题的例子。作为书写者，我们其实只在乎「我想要个一级标题，导出成任何格式时，这都要是个一级标题」，当你在 Ulysses 中输入标记时，Ulysses 就明白你想个一级标题。到导出成不同格式时，自动转换成各种文件格式中的一级标题。

其实分析了这么多，总结「360 度语义标记」对我们用户的实际意义就是：**可以在 Ulysses 中自己设计一种标记语法，并且导出为各类格式时，仍能保证样式正确**。

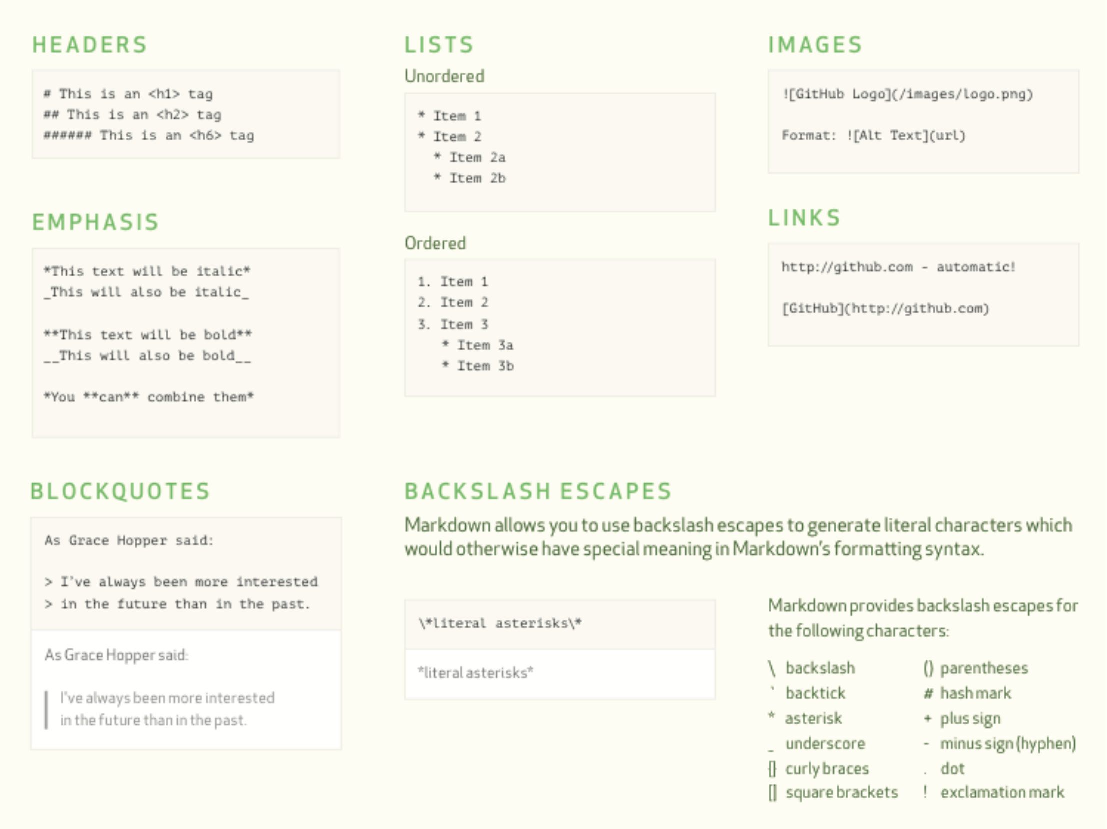
创建自定义语法

注：这一节需要对 Markdown 的各类语法有一定了解。如果你没有自定义语法的需求，或是刚接触 Markdown ，建议先跳过这一节。

当我们想**在 Ulysses 中书写、预览一种特殊的 Markdown 语法**时，就需要在 Ulysses 内新建自定义语法。和内置标记一样，自定义标记**只能用于 iCloud 文库中**的文稿。

第一章我们用普通话和方言的关系来比喻原生 Markdown 和衍生出的 Markdown 语法，GitHub Flavored Markdown（GFM）语法就是最好的例子。GitHub 自定义了这套语法，方便程序员编写项目的说明文档，上传给 GitHub 页面后会自动转成 HTML 网页。

而所谓自定义，就是为语法标记分配了不同的符号，本质上两者仍然是 Markdown 文件。接下来我们就以它为目标，尝试自定义一套 GFM 语法标记。



HEADERS

```
# This is an <h1> tag
## This is an <h2> tag
###### This is an <h6> tag
```

EMPHASIS

```
*This text will be italic*
_This will also be italic_

**This text will be bold**
__This will also be bold__

*You **can** combine them*
```

LISTS

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
2. Item 2
3. Item 3
   * Item 3a
   * Item 3b
```

IMAGES

```
![GitHub Logo](/images/logo.png)

Format: ![Alt Text](url)
```

LINKS

```
http://github.com - automatic!

[GitHub](http://github.com)
```

BLOCKQUOTES

```
As Grace Hopper said:

> I've always been more interested
> in the future than in the past.
```

As Grace Hopper said:

> I've always been more interested
> in the future than in the past.

BACKSLASH ESCAPES

Markdown allows you to use backslash escapes to generate literal characters which would otherwise have special meaning in Markdown's formatting syntax.

```
\*literal asterisks\*
```

literal asterisks

Markdown provides backslash escapes for the following characters:

\	backslash	()	parentheses
`	backtick	#	hash mark
*	asterisk	+	plus sign
_	underscore	-	minus sign (hyphen)
{}	curly braces	.	dot
[]	square brackets	!	exclamation mark

GFM 的一部分标记

首先在「偏好设置 – 标记」中，以 Markdown XL 为模版选择**新建标记**。

新建标记

通过与 GFM 官网列出的语法标记对比，发现有一部分的语法标记两者是完全相同的：

- 标题（一级标题至六级标题的标记完全一致）
- 粗体和斜体（虽然 GFM 能各用两种标记表示，但其实 Markdown XL 的标记都包含在内了）
- 有序列表
- 引用块
- 行内代码

这些语法标记不需要修改，直接保留即可。

可以修改的语法标记

对有一部分不同的语法标记，就需要手动修改。

在第一章中介绍过，Markdown 中的语法标记符号分为两种：第一种标记只需要在文字前加上即可符号（例如引用的 `>`），而第二种则需要在文字前后都加上字符（例如粗体的`**`）。在修改标记符号时，对标记的分类是不能改变的。

无序列表属于第一种，只能对应修改为 GFM 中的`*`，而删除线属于第二种，也只能对应修改为 `~~要删除的内容~~`。

双击标记的名称，就会弹出修改标记的窗口。**无序列表、删除**这两个符号。

修改标记

而**链接、图片**这两种标记则非常特殊。

从 Ulysses 中的标记和插入方式来看，似乎它们都不符合 GFM 中定义。但当我们将文稿从 Ulysses 中以 Markdown 的格式导出时，会发现导出的文稿中，链接和图片的标记都是与 GFM 一致的 `[]()` 和 `![]()`。这样的话，这种标记可以直接用于 GitHub，并不需要修改。

导出 Markdown 格式文稿

不过当我们直接在 Ulysses 书写时使用 `[]()` 和 `![]()` 时，会发现 Ulysses 不能正确识别它们。因为默认链接的标记被设定为`[]`，所以想输入 `[]()` 插入链接时，输入到一半（即 `[]` 部分），Ulysses 就会弹出插入链接窗口。那能不能将它修改为等我们在 `[]()` 中填好链接和文字后，再变成默认的显示样式呢?

无法按照 GFM 的习惯输入语法标记

很遗憾，这在 Ulysses 并不能实现。虽然我们可以修改标记符号，但是不能修改它的使用方式。在 Ulysses 中输入链接和图片的方式是先输入语法标记 `[]` 和 `(img)`，再弹出窗口填写链接，这和使用 `[]()` 的方式相冲突。

当然有一种曲线救国的办法，就是在 GFM 中删掉链接和图片的语法标记。在文中书写时，`[]()` 和 `![]()` 就会被当成普通文本，你就可以按照自己习惯的方式书写。这并不影响导出和之后在 GitHub 中使用，但是这也失去了预览链接和图片的效果。

所以我个人建议，大可不必在 Ulysses 中纠结链接和图片的语法标记，反正导出后是可以完美用于其他平台的。使用快捷键直接呼出弹窗，填入链接和图片，不比使用语法标记来的慢。

不可以修改的语法标记

我们可以按自己喜好设定语法标记，Ulysses 能理解我们想要的样式。但当 Ulysses 自身不能显示某些样式时，即使你在文中输入了相应的语法标记，它还是不能将它正确显示出来。这就是我们常说的「不支持某种语法」。

GFM 中规定了六种 Ulysses 不支持的语法：

- @：像在微信群聊中一样，可以直接呼叫特定用户
- #：链接到 Github 中特定的 issue 页面
- emoji：使用 `:sparkles:` 等字符直接输入 emoji
- 带语法高亮的代码块
- 待办清单
- 表格

即使你按照 GFM 的语法在 Ulysses 中输入，到了预览和导出的时候，Ulysses 还是不会明白，该如何呈现它们，只能直接将它们当作一般的纯文本。

在 Ulysses 中使用 GFM 的表格

反过来 Markdown XL 中也有许多语法标记是 GFM 没有的，例如注解、评论等。这些标记在 Ulysses 中能显示出来，但在 GitHub 中就没法被理解，也是直接当成纯文本对待了。对于这些标记，可以将它们从语法中删除。

删除语法标记

但是这并不意味着 Ulysses 不能写 GFM 或者其他语法。任何一门标记语言，本质都是纯文本，只是我们在用不同的工具，它们能区分文本其中标记的含义并加以显示。实际上，我们**还是可以在 Ulysses 任意写 GFM 中的标记**。文稿导出后，可以正确在 GitHub 中显示语法标记代表的功能。

文本完成后，使用 Ulysses 的「预览 」（快捷键 `⌘Command + ⇧Shift + P`）自然是不能正确的显示表格，不过我们可以借助 Marked 2 这款应用来预览。

Marked 2是一款专门用来预览Markdown文件的工具。安装Marked 2后，在Ulysses中选择「文本 – Markdown」，用Marked 2打开，选择GitHub样式，就能正确显示表格了。不过值得注意的是，Marked 2只负责预览，导出PDF等格式时，Ulysses还是无能为力的。

在Marked 2中预览表格

从预览GFM中的表格，和之前利用原始源插入HTML实现表格这两个例子可以看出，这类纯文本标记语言的使用可以非常灵活。我们可以在Ulysses中书写任意标记，之后导出纯文本放在合适的工具中使用即可。

至此我们就完成了对语法标记符号的设定，但是在iCloud文库中新建文稿时，还是会默认使用Markdown XL语法。点击「编辑 – 转换标记」，就可以选择自己想用的语法。

修改语法标记快捷键和样式

经过一番折腾，我们为各个语法标记设置了新的符号。别忘了还可以修改每一种**标记的配色和样式**。

「细线」和「黑暗」对应着正常模式和黑暗模式下，语法标记和字符的配色，最后一个则是打开打印机模式中突出显示时的颜色。对于每一种语法标记，都可以设置你喜欢的颜色。

修改语法标记颜色

默认设置下，Ulysses 没有**为所有语法标记设置快捷键**。在修改标记的弹窗中，可以手动为它们设定快捷键。

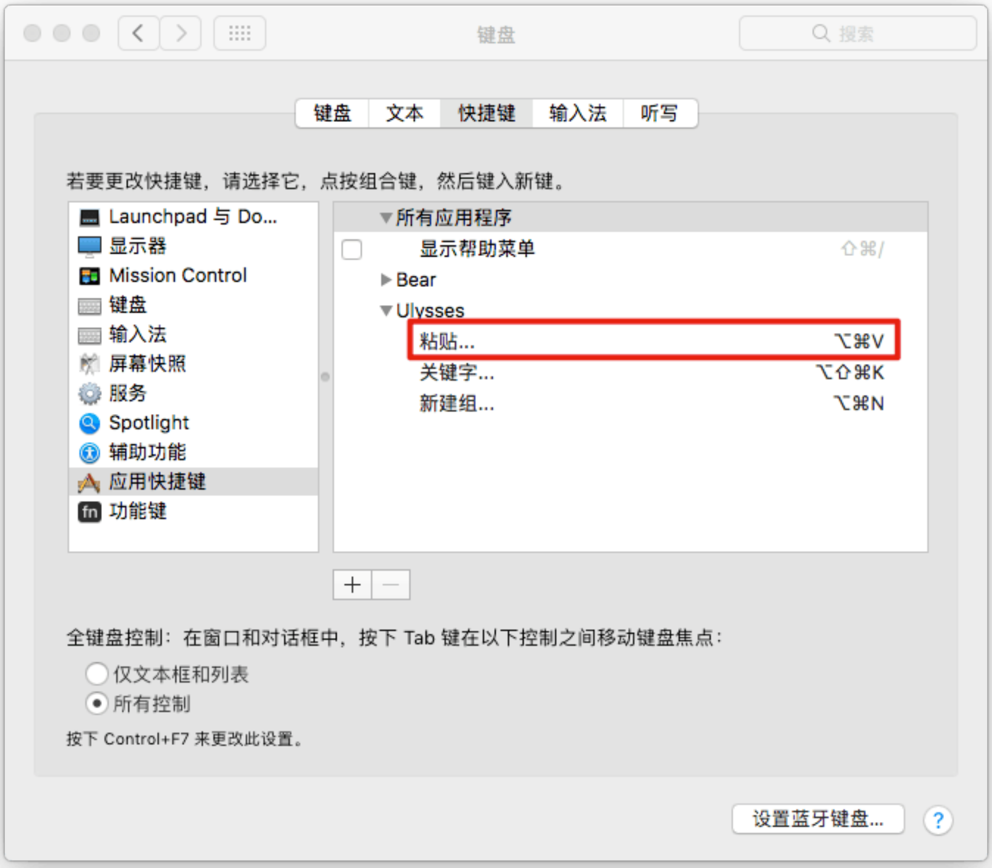
点击窗口中的「点击录制快捷键」，按下你想设定的键位，就完成快捷键设定。键位的设定是完全随你的喜好，不过如果你同时使用两款 Markdown 编辑器，就最好将语法标记快捷键保持一致。只有快捷键熟练到形成了肌肉记忆，才能提高输入的效率。如果两款编辑器的快捷键不一致，就会陷入每按一次之前，都要特地回忆一下的境地。

修改快捷键

因为我同时在使用 Bear 作为备忘录工具，这里我抛砖引玉，分享下自己统一这两款编辑器快捷键的思路。

1. 与Bear相同的语法标记，采用Bear中设定的快捷键；
2. Bear中没有的快捷键，按照「⌘Command + 标记英文单词首字母」的规律设定；
3. 如果与系统常用快捷键冲突，按照「⇧Shift + ⌘Command + 标记英文单词首字母」的规律设定。

第一类快捷键中，是各个编辑器（包括有些富文本编辑器）通用的快捷键设定，例如「粗体」（快捷键 `⌘Command + B`）、「链接」（快捷键 `⌘Command + K`）。第二类快捷键其实很少，因为通常按照「⌘Command + 标记英文单词首字母」的方式设定会与系统快捷键冲突，例如「视频（Video）」就不能设定成 `⌘Command + V`。而是按照第三类设定成 `⌘Command + ⇧Shift + V`。


键盘
搜索
键盘
文本
快捷键
输入法
听写
Launchpad 与 Do...
显示器
Mission Control
键盘
输入法
屏幕快照
服务
Spotlight
辅助功能
应用快捷键
功能键
+
—
设置蓝牙键盘...
?

若要更改快捷键，请选择它，点按组合键，然后键入新键。

▼所有应用程序
☐ 显示帮助菜单 ⇧⌘/
▶Bear
▼Ulysses
粘贴... ⌥⌘V
关键字... ⌥⇧⌘K
新建组... ⌥⌘N

全键盘控制：在窗口和对话框中，按下 Tab 键在以下控制之间移动键盘焦点：

○ 仅文本框和列表
● 所有控制

按下 Control+F7 来更改此设置。

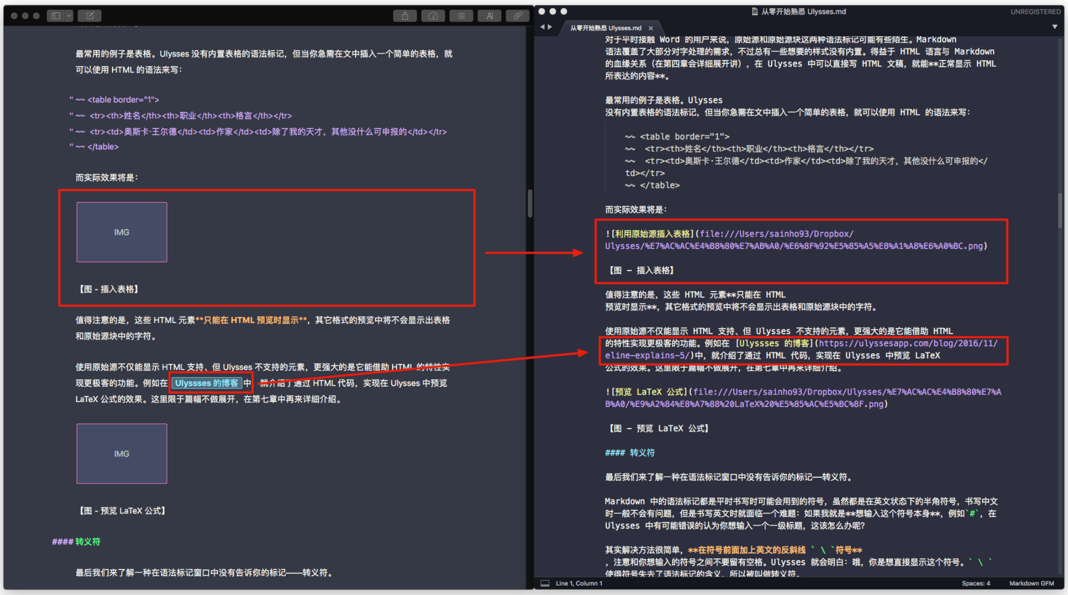
在系统设定中修改快捷键

不过 Ulysses 默认设置 `⌘Command + ⇧Shift + V` 为「智能粘贴」的快捷键。我认为 `⇧Shift` 作为修饰键，含义应该是「切换原有快捷键的功能」，而 `⌥Option` 则是「类似原有快捷键的另一种功能」。所以我仍在 Ulysses 中将 `⌘Command + ⇧Shift + V` 分配给「插入视频」，而利用系统的快捷键修改功能将 `⌘Command + ⌥Option + V` 分配给「智能粘贴」。

导入导出时的语法标记转换

自定义标记解决的需求场景是「在 Ulysses 中使用其他语法」，当我们不得不**在 Ulysses 外**编辑使用文稿时，就必须注意文稿导入导出时的语法标记转换。

首先可以明确的是，**外部文件夹中的文稿**只能使用 Markdown 语法，于是不存在标记转换的问题。使用其他工具，例如 Byword，打开并修改了文稿，只要使用的是原生 Markdown 语法，即使是脚注、链接、图片等特殊的标记，Ulysses 中都能正确显示。



最常用的例子是表格。Ulysses 没有内置表格的语法标记，但当你急需在文中插入一个简单的表格，就可以使用 HTML 的语法来写：

```
'' ~~ <table border="1">
'' ~~  <tr><th>姓名</th><th>职业</th><th>格言</th></tr>
'' ~~  <tr><td>奥斯卡·王尔德</td><td>作家</td><td>除了我的天才，其他没什么可申报的</td></tr>
'' ~~ </table>
```

而实际效果将是：

【图 - 插入表格】

值得注意的是，这些 HTML 元素**只能在 HTML 预览时显示**，其它格式的预览中将不会显示出表格和原始源块中的字符。

使用原始源不仅能显示 HTML 支持、但 Ulysses 不支持的元素，更强大的是它能借助 HTML 的特性实现更极客的功能。例如在 Ulyssses 的博客 中，就介绍了通过 HTML 代码，实现在 Ulysses 中预览 LaTeX 公式的效果。这里限于篇幅不做展开，在第七章中再来详细介绍。

【图 - 预览 LaTeX 公式】

转义符

最后我们来了解一种在语法标记窗口中没有告诉你的标记——转义符。

```
对于平时接触 Word 的用户来说，原始源和原始源块这两种语法标记可能有些陌生。Markdown
语法覆盖了大部分对字处理的需求，不过总有一些想要的样式没有内置。得益于 HTML 语言与 Markdown
的血缘关系（在第四章会详细展开讲），在 Ulysses 中可以直接写 HTML 文稿，就能**正常显示 HTML
所表达的内容**。

最常用的例子是表格。Ulysses
没有内置表格的语法标记，但当你急需在文中插入一个简单的表格，就可以使用 HTML 的语法来写：

    ~~ <table border="1">
    ~~  <tr><th>姓名</th><th>职业</th><th>格言</th></tr>
    ~~  <tr><td>奥斯卡·王尔德</td><td>作家</td><td>除了我的天才，其他没什么可申报的</
    td></tr>
    ~~ </table>

而实际效果将是：

![利用原始源插入表格](file:///Users/sainho93/Dropbox/
Ulysses/%E7%AC%AC%E4%B8%80%E7%AB%A0/%E6%8F%92%E5%85%A5%E8%A1%A8%E6%A0%BC.png)

【图 - 插入表格】

值得注意的是，这些 HTML 元素**只能在 HTML
预览时显示**，其它格式的预览中将不会显示出表格和原始源块中的字符。

使用原始源不仅能显示 HTML 支持、但 Ulysses 不支持的元素，更强大的是它能借助 HTML
的特性实现更极客的功能。例如在 [Ulyssses 的博客](https://ulyssesapp.com/blog/2016/11/
eline-explains-5/)中，就介绍了通过 HTML 代码，实现在 Ulysses 中预览 LaTeX
公式的效果。这里限于篇幅不做展开，在第七章中再来详细介绍。

![预览 LaTeX 公式](file:///Users/sainho93/Dropbox/Ulysses/%E7%AC%AC%E4%B8%80%E7%A
B%A0/%E9%A2%84%E8%A7%88%20LaTeX%20%E5%85%AC%E5%BC%8F.png)

【图 - 预览 LaTeX 公式】

#### 转义符

最后我们来了解一种在语法标记窗口中没有告诉你的标记—转义符。

Markdown 中的语法标记都是平时书写时可能会用到的符号，虽然都是在英文状态下的半角符号，书写中文
时一般不会有问题，但是书写英文时就面临一个难题：如果我就是**想输入这个符号本身**，例如`#`，在
Ulysses 中有可能错误的认为你想输入一个一级标题，这该怎么办呢？

其实解决方法很简单，**在符号前面加上英文的反斜线 ` \ `符号**
，注意和你想输入的符号之间不要留有空格。Ulysses 就会明白：哦，你是想直接显示这个符号。` \ `
使得符号失去了语法标记的含义，所以被叫做转义符。
```

Ulysses 内外文稿语法标记对比

其次我们来探究**从 iCloud 文库中导出 Markdown 文稿**的情况。在导出时，Ulysses 会按照你设定的语法标记符号导出。如果是原生 Markdown 中不包含的语法标记，例如注解、评论等，就会不被导出。

最后是**需要打开 Ulysses 外部文件**的场景。之前我们一直都默认由 Ulysses 作为创建文稿，但实际上 Ulysses 也能像其他编辑器一样，在不将它列入外部文件夹的前提下，**直接打开**由其他编辑器创建的 Markdown 文稿。

不过这种操作在我看来「十分危险」，因为 **Ulysses 会修改某些语法标记**。例如在文稿中使用行内式的链接写法，使用 Ulysses 修改内容后会变成参考式的写法。

用 Ulysses 打开文稿

所以如果需要打开他人创建的文稿，最好是直接**将文稿导入到 Ulysses 中**，而不是直接打开。导入 Ulysses 后，相当于在 Ulysses 中创建了一个副本，可以随意修改。原件保留在 Ulysses 外以备不时之需。如果是由原生 Markdown 语法书写的文稿，导入后是可以正确显示脚注、链接等内容的。

导入文稿到 Ulysses

灵活组合 Ulysses 的功能来辅助写作

Ulysses 在功能设计上非常注意从细节出发，而不是包装出一个新的概念或者功能。落实在具体工具上，就会发现 Ulysses 的功能，都是根据写作时的真实需求来设计的。

「快速打开」与「语义搜索」

「快速打开」是一项能节省我们不少操作时间的功能。

在写作《OmniFocus 入门与进阶》时，当提到某个具体的功能，常常需要查看下前几章中对它的描述，以免内容写重复了。但是我写作时通常会使用「仅限编辑器」视图，为了看下另外一章，只能切换成文库模式，再找到文章中的内容，最后还要切换到当前写的文章，整个流程严重打断了之前写作的思路。

而通过「快速打开」，我可以一步打开想要的文章。

快速打开窗口

使用快捷键 `⌘Command + O`，就能打开「快速打开」窗口，默认会显示最近编辑过的文章。当然最主要是通过搜索找到文章。想搜索一篇文章时，在「快速

打开」窗口输入标题或者文中出现的关键词句，包含相关内容的文章就会显示出来。

语义搜索

如果「快速打开」的搜索结果你还觉得不够精准，Ulysses 更是提供一种**「语义搜索」**。在文稿栏的上方选择要搜索内容的所带的标记，例如是粗体，或是标题。这样当你的文库非常大时，能有效地缩小搜索的范围，免得搜出几十条结果要一条条筛选。不过由于语义搜索是非常需要计算性能的操作，它的搜索范围被限定在了当前组。

活用「快速打开」，还能「制作」出新的工具。

Scrivener 有一个特色功能——「快速参考窗口」。其实本质上就是将文库中的一篇文章在新窗口打开，这样在写作时可以方便的查看素材或者别的文稿。

Scrivener 中的快速参考窗口

在 Ulysses 中，通过「快速打开」和「多窗口（标签）」的特性，也能拥有一个快速参考窗口。

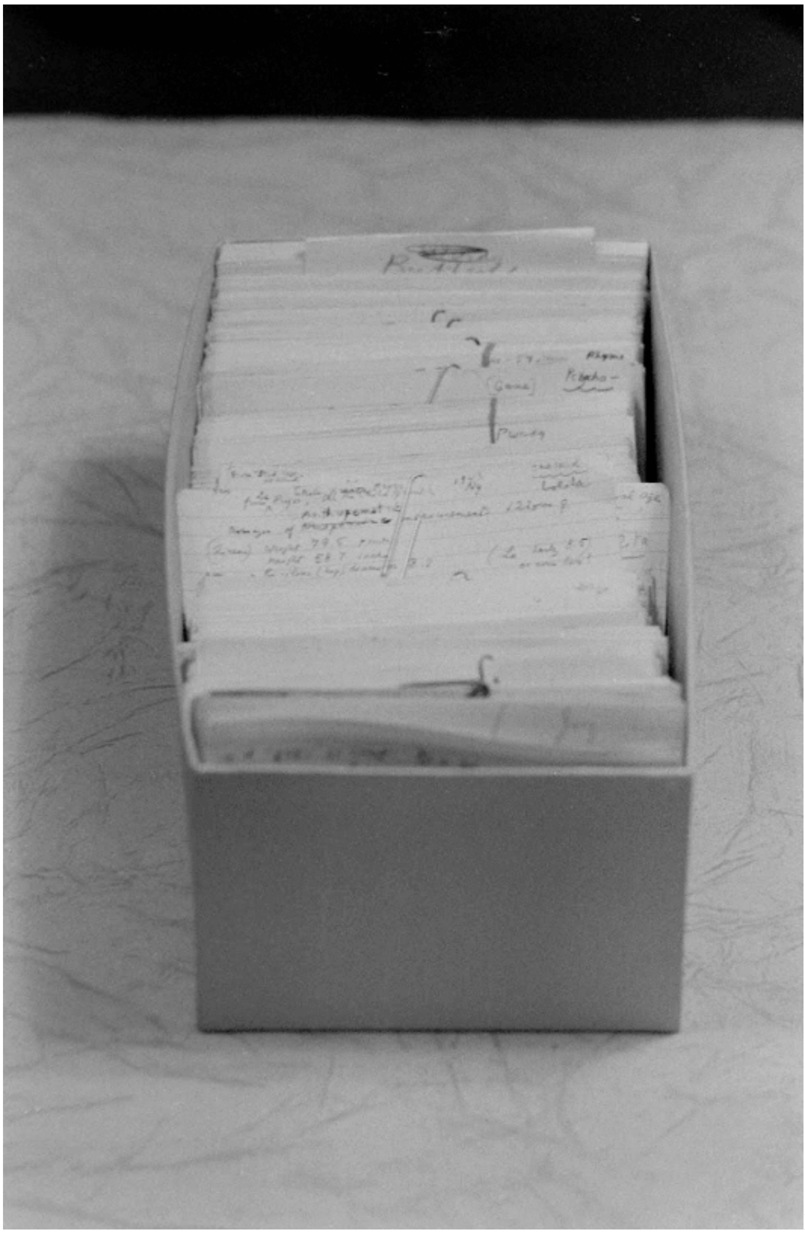
第一步当然是**创建一个新的窗口**，点击「文件 – 新窗口」（快捷键 `⌘Command + ⌥Option + N`），就能呼出新窗口。然后使用快捷键 `⌘Command + O`，就能**打开「快速打开」窗口**，默认会显示最近编辑过的文章。当然在窗口中也可通过搜索找到文章。

Ulysses 中的快速参考

通过两个功能的结合，就实现了所谓的「快速参考窗口」。

在 Ulysses 中实践卡片式写作

另一个可以从 Scrivener 中「偷师」的理念，就是「卡片式写作」。

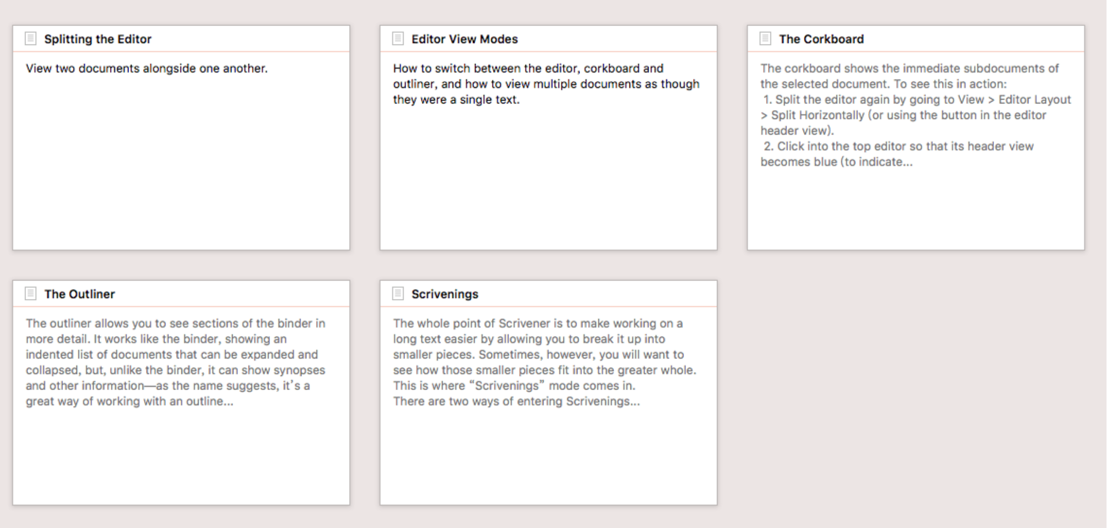


纳博科夫的卡片

践行卡片式写作最著名的当属昆虫学家、小说《洛丽塔》的作者纳博科夫。他写小说时，不是在文稿本上长篇累牍，而是在一张张卡片上不按顺序写作：

> 我随意在空白处填写。我在卡片上写下这些段落，直到完成整部小说，我的写作计划是灵活的，但我对写作工具则相对讲究：横格的布里斯托卡片、削得不太尖的橡皮铅笔。
>
> —— 纳博科夫 1967 年接受《巴黎评论》的采访

卡片在阅读时可以方便地收集整理素材，写作时只关注于庞大小说中的单点情节，而修改时可以随意地再次组织文本。在 Scrivener 中，提供了专门的公告板视图，让文稿显示成一张张卡片，随意移动重组卡片的顺序，还可以在卡片表面上写下备注，就像真实地卡片一样。



Splitting the Editor

View two documents alongside one another.

Editor View Modes

How to switch between the editor, corkboard and outliner, and how to view multiple documents as though they were a single text.

The Corkboard

The corkboard shows the immediate subdocuments of the selected document. To see this in action:
1. Split the editor again by going to View > Editor Layout > Split Horizontally (or using the button in the editor header view).
2. Click into the top editor so that its header view becomes blue (to indicate...

The Outliner

The outliner allows you to see sections of the binder in more detail. It works like the binder, showing an indented list of documents that can be expanded and collapsed, but, unlike the binder, it can show synopses and other information—as the name suggests, it's a great way of working with an outline...

Scrivenings

The whole point of Scrivener is to make working on a long text easier by allowing you to break it up into smaller pieces. Sometimes, however, you will want to see how those smaller pieces fit into the greater whole. This is where "Scrivenings" mode comes in.
There are two ways of entering Scrivenings...

Scrivener 的公告板

在 Ulysses 虽然没有专门的卡片视图，但其实也完全可以实现类似的效果。

在 Ulysses 中的文稿栏中，每一份文稿都可以看做一张独立的卡片。如果写作时发现一篇文稿的内容涉及多个主题了，应该分成多张卡片，将光标移到想**拆分卡片**的地方，点击「编辑 – 选择时分断」（快捷键 `⇧Shift + ⌘Command + B`），就能将一张卡片分成两张。

拆分文稿

相反，如果认为两张卡片其实写完整地叙述了同一个主题，应该是写在一张卡片上的内容，就可以将两张卡片**合并**成一张。点击「文件 – 合并文稿」，Ulysses 就会将生成一份新的文稿。

合并文稿

与合并文稿相类似的是**接合文稿**功能。选中两份文稿后点击「文件 – 接合文稿」（快捷键 `⌘Command + J`），不会产生新的文稿，而是将两份文稿会一起显示，中间以分割线划分。移动文稿时，两份文稿也是一起移动。如果两张卡片有着强烈的上下文联系，就应该选择接合而不是合并文稿。

接合文稿

在卡片式写作中必不可少的是给卡片写备注功能。无论是写作还是后期修改时，都不应该去读卡片的内容，才知道其中写了什么。在 Ulysses 的附件视图中，可以给文稿添加**注释**，简明地写清这份文稿的大致内容。后期修改时就可以查看注释来了解卡片。值得一提的是，注释功能只有在 iCloud 文库中才能使用。

给文章添加注释

能方便地拆分和接合文稿，又能给文稿加注释，虽然没有专门提出卡片式写作的概念，其实 Ulysses 在功能上已经完全能够实现了。

在 Ulysses 中写作时的提速技巧

Ulysses 对写作体验的优化，细致到了使用时的每一个细节。

在写长文章中，虽然可以用快捷键位移光标，但通常会习惯使用鼠标来回移动。而在 Ulysses 中，可以使用更快速的在文中跳转的方式。

首先点击「视图 – 显示段落号」，然后双击右侧的段落号，就可以在这一段设定一个书签。之后在导航窗口中（快捷键 `⌘Command + 8` ），就会发现除了

各级标题，还有你自己添加的书签。利用书签在文中跳转，要比键盘移动光标和鼠标滚动来的精确的多。

给文章添加书签

书写Markdown文本的一大便利，就是可以大大减少右手在键盘和鼠标之间的切换。而在Ulysses中，更应该多使用快捷键而不是鼠标，去提高使用时的速率。乍一看菜单栏中很多功能都没有设置快捷键，其实在macOS系统中都可以自定义。打开「系统偏好设置 – 键盘 – 快捷键 – 应用快捷键」，可以给菜单栏中的功能设置快捷键。

给Ulysses添加快捷键

但其实Ulysses对快捷键的支持还远不止菜单栏，还有许多隐藏功能只有用快捷键才能使用。打开Ulysses的介绍文档，有一章专门列出了Ulysses的快捷键，如果感兴趣可以深入阅读。

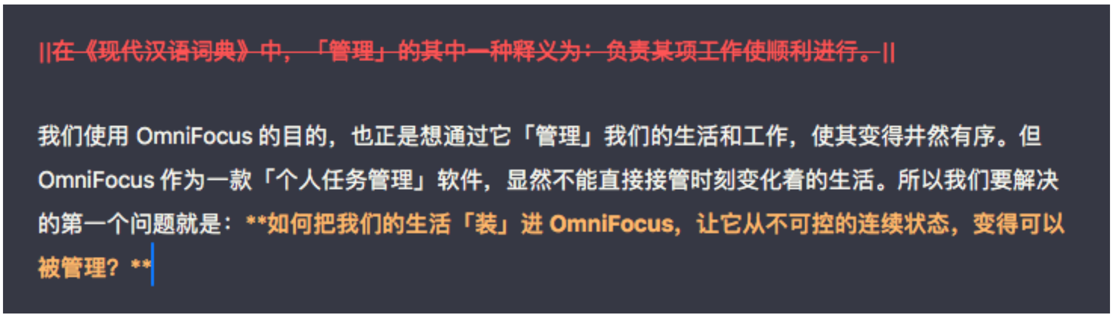
在Ulysses中编辑文章

海明威曾说：

所有的初稿都是垃圾。

初稿完成后的润色，是任何文章发表前必不可少的一步。至于具体如何修改，每个作者都有自己的习惯，但最基本的必然有**删减和批注**两种方式。

删除标记是 Ulysses 中特别支持的功能，所以只有在 iCloud 文库中的文稿才能使用。选中文字后点击「标记 – 删除」，文字就会以特别显眼的方式呈现，而且不会被导出。



||~~在《现代汉语词典》中，「管理」的其中一种释义为：负责某项工作使顺利进行。~~||

我们使用 OmniFocus 的目的，也正是想通过它「管理」我们的生活和工作，使其变得井然有序。但 OmniFocus 作为一款「个人任务管理」软件，显然不能直接接管时刻变化着的生活。所以我们要解决的第一个问题就是：**如何把我们的生活「装」进 OmniFocus，让它从不可控的连续状态，变得可以被管理？**

删除标记的效果

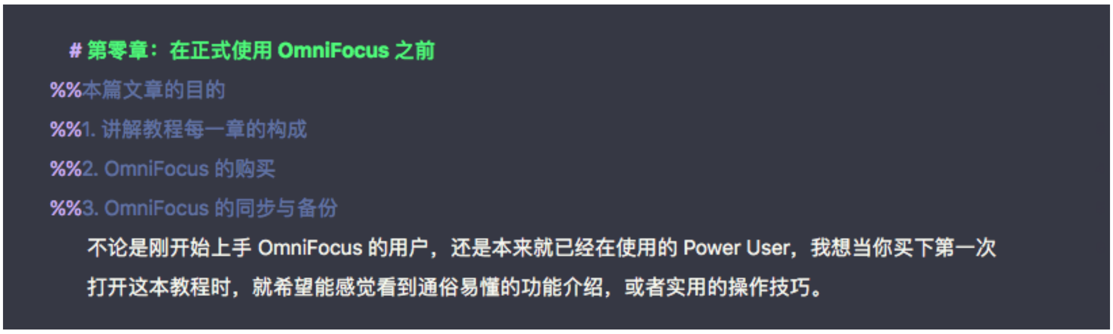
可能你会问，为什么要使用删除标记，而不是直接将文字删除呢。之前我们介绍过 Ulysses 的版本历史功能，其实是不担心直接删除后又想找回的情形的。

就我个人认为，我们在修改时，其实对文章的某些段落修改前后的效果很难有直接判断，怎么判断到底怎么写好呢？最直接的当然是放在一起对比，此时使用删减标记，既能明显地区分修改前后的段落，又能直接进行对比。

修改润色文章，有时也很难找到足够的时间一次性完成。又或者有的作者希望先一口气审阅完初稿，再回头专注修改。这时就需要在文中需要修改的段落，做上批注提醒自己之后留意。Ulysses 提供了两种方式在编辑时做批注：**评论和注解**。

添加注解和评论

两者相同之处在于，所批注的内容都不会被导出。而不同的是在文中的插入方式。评论是直接在文中输入一段，而注解需要先选中文稿中的一段文字，特别对这段文字进行批注。所以**用评论适合添加一些不针对特定内容的批注**，例如「上下两段衔接的不连贯，加一段承上启下的段落」；而**注解则适合对某个词或句子添加批注**，例如「这个长句拆成短句，去掉一些无用的连词」。



```
   # 第零章：在正式使用 OmniFocus 之前
%%本篇文章的目的
%%1. 讲解教程每一章的构成
%%2. OmniFocus 的购买
%%3. OmniFocus 的同步与备份
    不论是刚开始上手 OmniFocus 的用户，还是本来就已经在使用的 Power User，我想当你买下第一次
    打开这本教程时，就希望能感觉看到通俗易懂的功能介绍，或者实用的操作技巧。
```

添加注释块

而注释块其实更应该叫做「评论块」，就好像「代码」和「代码块」的区别，评论是在行内添加一句，而注释块则必须是单独的一段。除此之外用法完全一致。

搜索和替换

在编辑过程中，如果需要一次性修改某个单词，「搜索&替换」就非常实用。和之前介绍过的文稿栏中的搜索不同，按快捷键 `⌘Command + F` 就能在编辑界面唤出「搜索&替换」界面。而搜索的范围就局限在当前文稿中，而不是整个文库。

用过滤器整理文库

经过素材收集、写作和编辑阶段，文章也初见雏形。不过你若回头看看自己的文库，可能就会发现它已经被长长短短的素材和文稿塞满。如果不加以整理，势必要影响使用的效率。

而在 Ulysses 这样一个以文件夹形式组织的文库中，最便捷就是使用过滤器进行整理。

设定关键字

过滤器就像其他软件中的「智能文件夹」，通过设定筛选条件，将符合条件的文稿放在一起。而在 Ulysses 中，可以利用文本、关键字、日期三类信息设定筛选条件。相比文本和日期这类不方便记忆的信息，关键字无疑是最合适当作筛选条件的。

打开附件窗口（快捷键 `⌘Command + 4` ），就能给文稿添加「关键字」，其实就是其他应用中常见的「标签」的概念。

添加关键字

该写怎么样的关键字一直是整理文库的关键。在这一方面我想每个人都有自己的习惯，这里介绍一下我自己的用法以抛砖引玉。我使用的关键字主要分为两类：表示**文章的状态**或**文章的内容**。对于一份文稿，无非会经历以下几个阶段：

- 初稿中：文章尚未完成；
- 待编辑：初稿完成需要修改；

- 已发布：已经在少数派发布；
- 暂不发布待修改：达不到发布标准，日后文章经过修改后再提交少数派后台审核。

对于文稿，我就以这四类关键字进行区分，在 Ulysses 中创建相应的过滤器，这样即使积累了很多文稿，也能快速找出要继续写或者修改的稿子。

而对于素材，我更倾向于根据内容来划分，例如「个人用法」、「功能讲解」或者「参考链接」等。

设定过滤器

设定好关键字后就可以继续设定筛选条件，以创建过滤器。如果关键字设定的合理，只需要简单的选择一个关键字作为条件即可。以我自己的使用情况为例，就利用关键字设定了以下过滤器：

- 写作中的文章：关键字为「草稿」，意味着文章还没有写完；
- 待编辑的文章：关键字为「编辑」，意味着文章初稿完成需要修改；
- 未发布：关键字为「未发布」，被退稿需要大篇幅修改的文章；

- 已发布：关键字为「已发布」，已经发布可以归档的文章。

各个过滤器对应着不同的文稿状态。当我想找去开始修改文章，给文章配图时，直接点进「待编辑的文章」即可。当然如果你有需要，也可以把过滤器设置的更复杂一些。

新建过滤器

在整理文库的过程中不可避免要对文稿进行排序，在文稿栏的上方就可以操作。这样整理过后的文库，不仅条理清晰，而且方便日后写作时，能最快找到要打开的文稿。

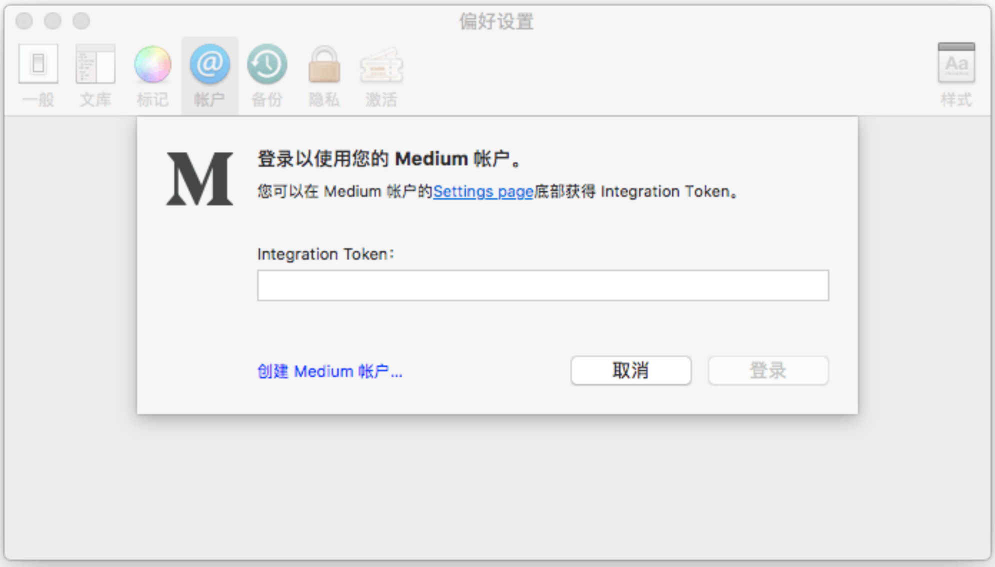
在组中排序文章

直接发布到 Medium 和 Wordpress

第一章中介绍过将 Ulysses 中的文稿以各种格式导出，其实 Ulysses 还可以直接发布到 Medium 和 Wordpress 网站上。

Medium 是 Twitter 联合创办人埃文·威廉姆斯和比兹·斯通创办的一家新型博客网站。因为威廉姆斯的背景，Medium 上聚集了一篇相当有深度的作者，特

别是在科技行业。而连接 Ulysses 和 Medium 的方法也十分简单，在 Medium 的设置界面中将个人 token 复制到 Ulysses 中即可。


偏好设置
一般
文库
标记
帐户
备份
隐私
激活
样式
M
创建 Medium 帐户...
取消
登录

登录以使用您的 Medium 帐户。

您可以在 Medium 帐户的Settings page底部获得 Integration Token。

Integration Token:

登陆 Medium 账号

Wordpress 则是更加为人熟知的建站工具了。利用 Wordpress 建站有两种方式：

1. 第一种是像博客一样，直接在 Wordpress 网站上建站，对应 Ulysses 中的「Wordpress.com」；
2. 第二种则是利用 Wordpress 的开源博客软件，自己购买域名服务器来建站，对应 Ulysses 中的 「自定义 Wordpress」。

限于篇幅，这里不展开讲 Wordpress 建站的相关内容，有兴趣的可以去 Wordpress 官网了解。

小结

经过这两章的介绍，Ulysses for Mac 的主要功能都已经展示过了。我们两次顺着从写作前准备到写作后发布的流程，由浅入深地介绍了 Ulysses 的相关功能。之所以采取这样的讲解逻辑，是希望你不是简单的了解某个功能，而是知晓这个功能该在写作的什么环节发挥怎样的作用。

那么，同样的从写作前准备到写作后发布的需求，在 Ulysses for iOS 上该如何实现？受限于 iOS 设备的硬件，Ulysses for iOS 的作用会被削弱吗？这些问题我们将在下一章解答。

我们下一章见！